

Figure 1

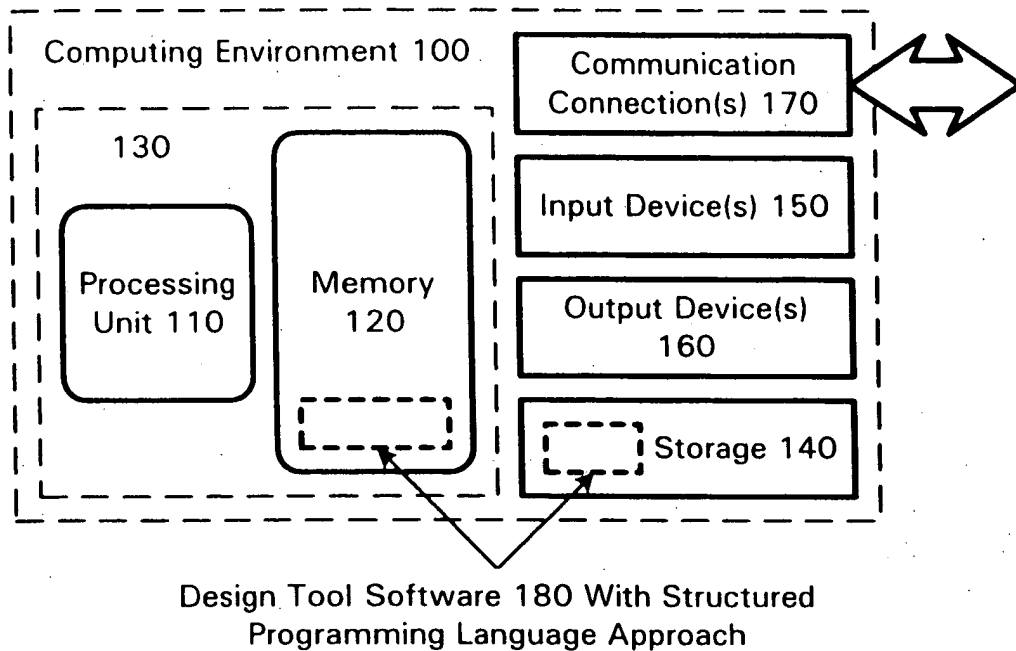


Figure 2

```
200  int gcd(int input1, int input2) {  
    ↓  if ( input1 == 0 || input2 == 0 )  
        return 0;  
    while ( input1 != input2 ) {  
        if ( input1 < input2 )  
            input2 -= input1;  
        else  
            input1 -= input2;  
    }  
    return input1;  
}
```

Figure 10

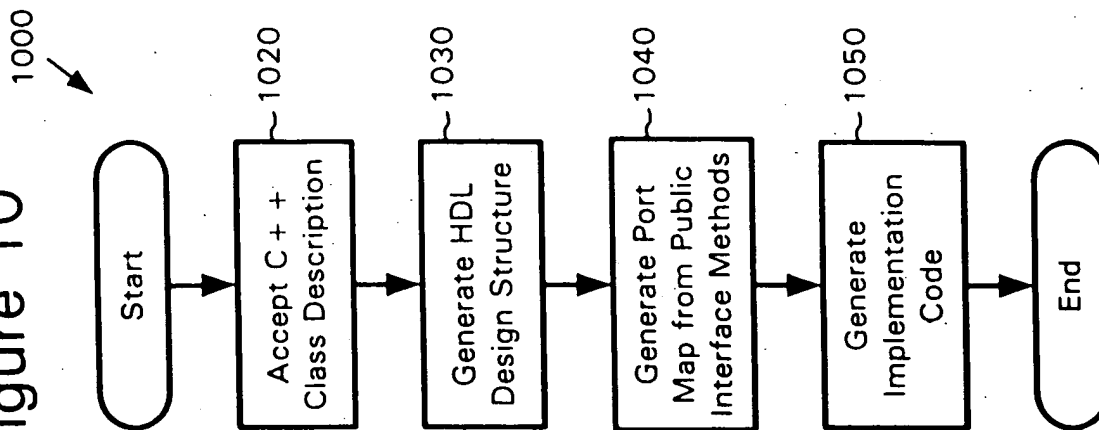


Figure 5

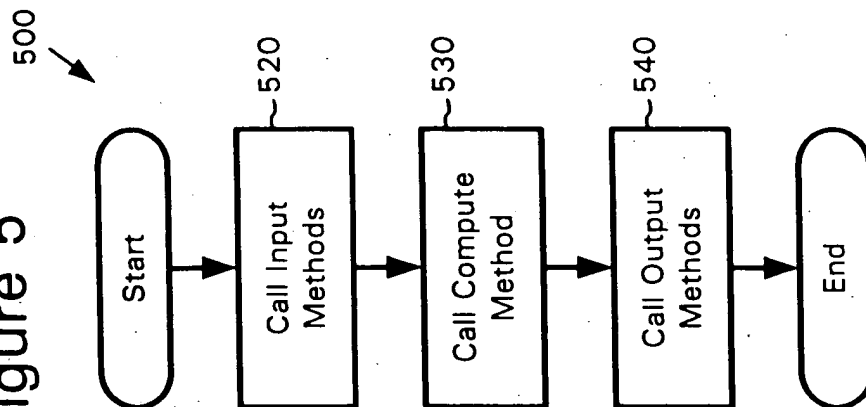


Figure 3

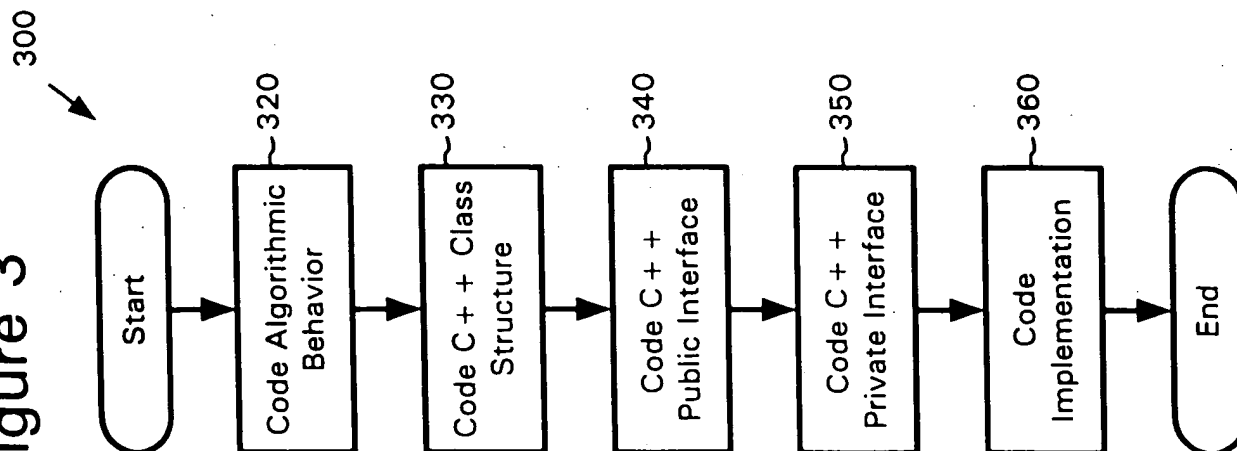


Figure 4

```
class GCD {  
405  [ #pragma design  
    public:  
        // Input Ports  
        void input1( int input ) { d_input1 = input; }  
        void input2( int input ) { d_input2 = input; }  
410  [  
        // Process compute  
        void compute( void );  
414  [  
        // Output Ports  
        int output( void ) { return d_output; }  
416  [  
    private:  
        int d_input1;  
        int d_input2;  
        int d_output;  
430  [ ];  
    void GCD::compute( void ) {  
        d_output = 0;  
        if ( d_input1 && d_input2 ) {  
            while ( d_input1 != d_input2 ) {  
                if ( d_input1 < d_input2 )  
                    d_input2 -= d_input1;  
                else  
                    d_input1 -= d_input2;  
            }  
            d_output = d_input1;  
        }  
450  [ ]  
    }  
}
```

400

Figure 6

```
610  GCD gcd1;
    ...
    void Top::compute( void ) {
    ...
    gcd1.input1( d_abc + 23 );
    gcd1.input2( 46 );
    gcd1.compute();
    common_denom = gcd1.output();
    ...
    }
```

600

622

624

626

Figure 8

```
class Testbench {
public:
    void test( void );
private:
    GCD4 gcd;
};

void Testbench::test( void ) {
    static int vec[20] = {
        300, 21, 96, 3681,
        10, 0, 2510, 1111,
        4325, 90100, 50, 275,
        12, 18, 60, 300,
        1, 10, 100, 1000
    };

    for ( int i=0; i <= (20-4); ++i ) {
        gcd.input( &vec[i] );
        gcd.compute();
        cout << "GCD(" << vec[i] << "," << vec[i+1] << ","
            << vec[i+2] << "," << vec[i+3] << ") = "
            << gcd.output() << endl;
    }
}
```

800

810

820

830

Figure 7

```
class GCD4 {
public:
    // Input Ports - ( memory or stream )
    void input( int input[] ) { d_input = input; }
    // Process compute
    void compute( void );
    // Output Ports
    int output( void ) { return d_output; }
private:
    // Internal algorithm
    int get_gcd( int i, int in1, int in2 );
    int *d_input;
    int d_output;
    GCD gcd[2];
};

int GCD4::get_gcd( int i, int in1, int in2 ) {
    gcd[i].input1( in1 );
    gcd[i].input2( in2 );
    gcd[i].compute();
    return gcd[i].output();
}

void GCD4::compute( void ) {
    d_output = 0;

    int tmp_out1 = get_gcd( 0, d_input[0], d_input[1] );
    int tmp_out2 = get_gcd( 1, d_input[2], d_input[3] );

    tmp_out1 = get_gcd( 0, tmp_out1, tmp_out2 );

    d_output = tmp_out1;
}
```

Diagram annotations:

- 700 points to the `class GCD4 {` line.
- 710 points to the `public:` section.
- 712 points to the `void input(int input[]) { d_input = input; }` line.
- 720 points to the `private:` section.
- 722 points to the `int get_gcd(int i, int in1, int in2);` line.
- 724 points to the `GCD gcd[2];` line.
- 730 points to the `int GCD4::get_gcd(int i, int in1, int in2) {` line.
- 740 points to the `void GCD4::compute(void) {` line.

Figure 9

900



```
struct bitvector4 {  
    #pragma builtin  
    signed int val : 4;  
    bitvector4 () {}  
    bitvector4 ( signed int n ) { val = n; }  
    signed int operator = ( signed int n ) { val = n; return val; }  
    operator signed int () const { return val; }  
    signed int operator -= ( signed int n ) { val -= n; return val; }  
    signed int operator += ( signed int n ) { val += n; return val; }  
    signed int operator *= ( signed int n ) { val *= n; return val; }  
    signed int operator /= ( signed int n ) { val /= n; return val; }  
    signed int operator ++ () { return ++val; }  
    signed int operator ++ ( int ) {  
        bitvector4 tmp = val;  
        ++val;  
        return tmp;  
    }  
    signed int operator -- () { return --val; }  
    signed int operator -- ( int ) {  
        bitvector4 tmp = val;  
        --val;  
        return tmp;  
    }  
};
```

1100

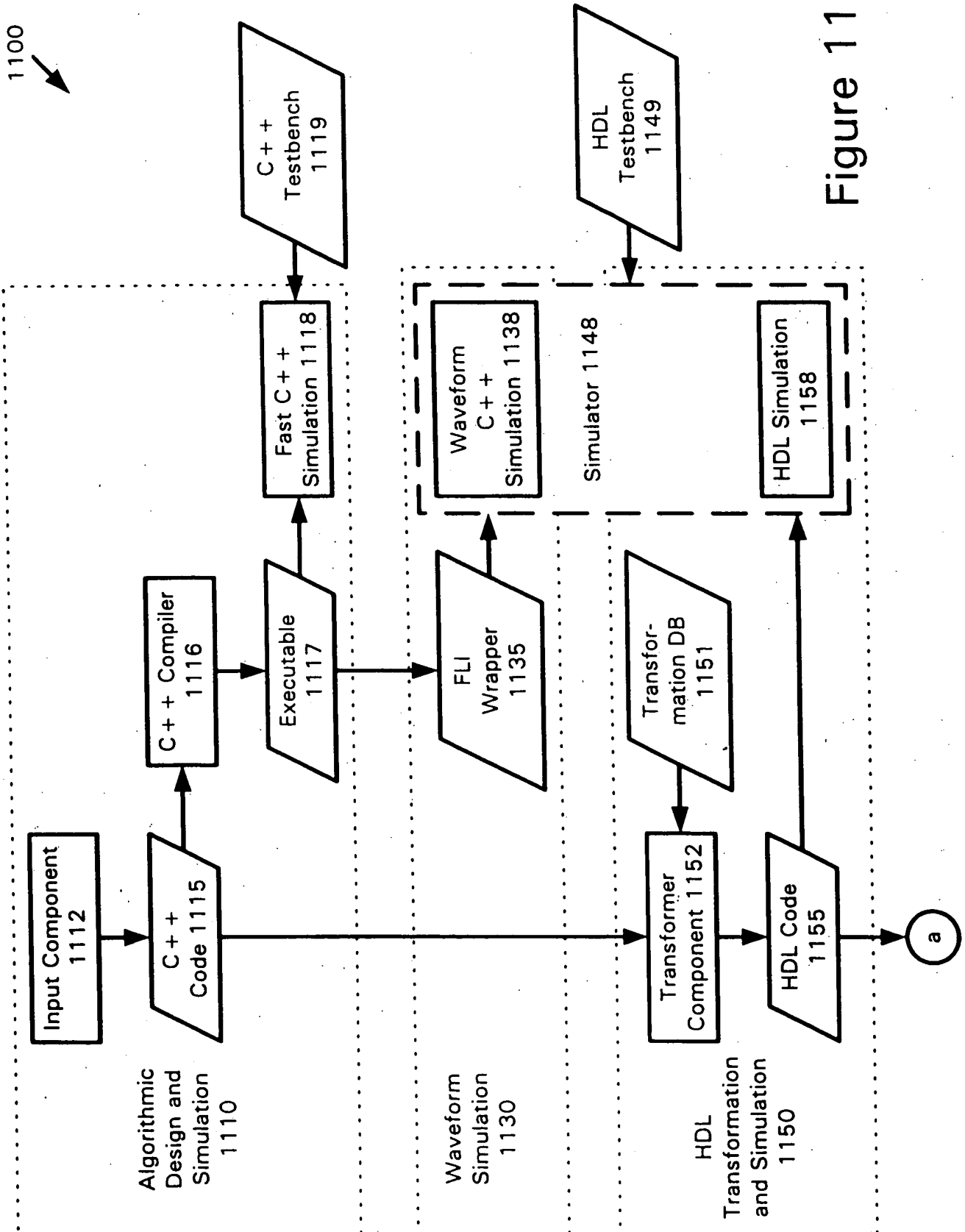


Figure 11a

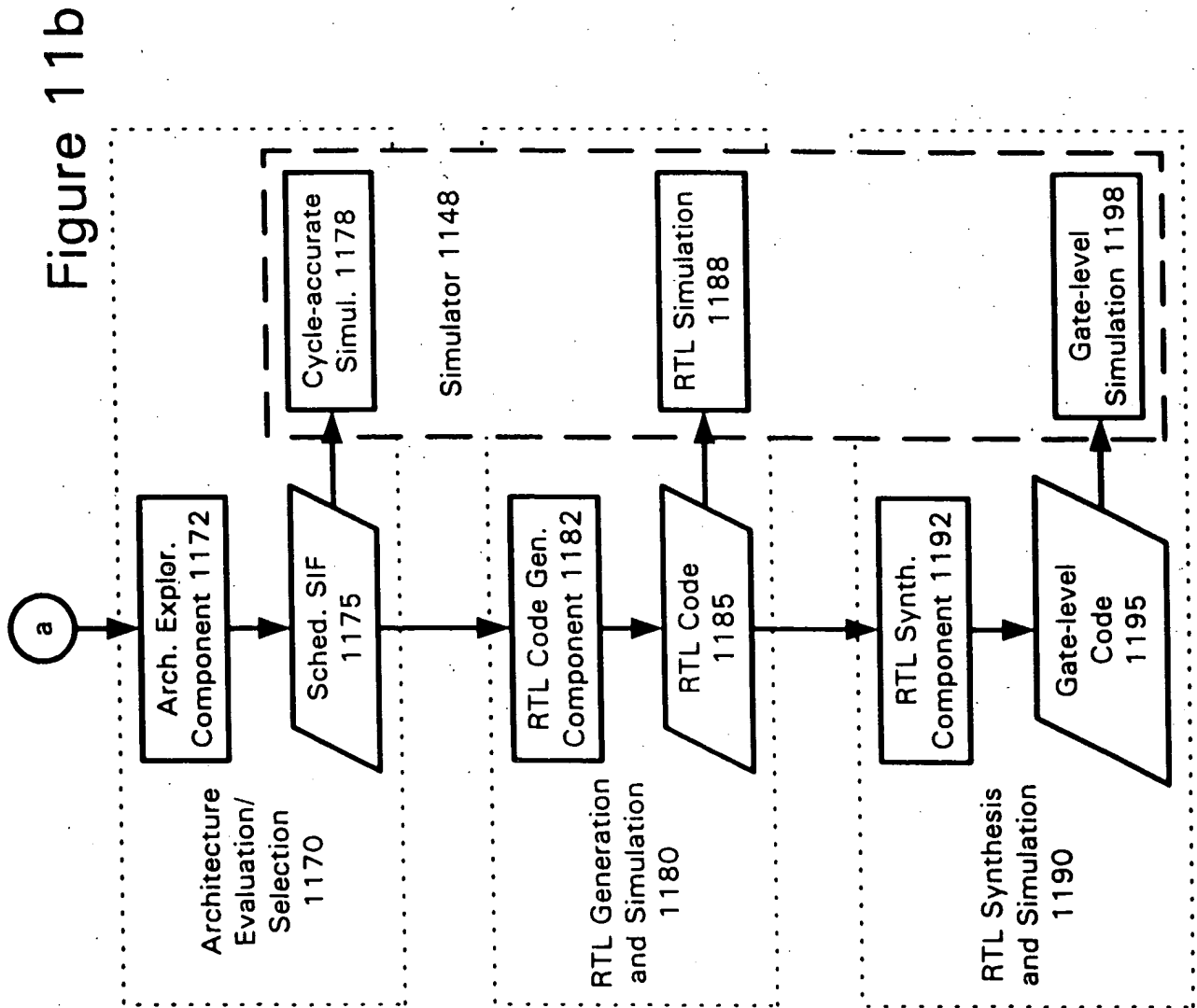


Figure 12

1200


| Benchmark | GCD | | FIR | | 5 th Order Filter | | JPEG | |
|--|------------|---------------|------------|---------------|------------------------------|--------------|-------------|--------------|
| original C++ (C++ testbench) | 25 lines | 0.67 seconds | 68 lines | 0.75 seconds | 77 lines | 0.95 seconds | 257 lines | 0.29 seconds |
| original C++ w/FLI (VHDL testbench) | 127 lines | 11.8 seconds | 221 lines | 2.5 seconds | 230 lines | 2.88 seconds | 415 lines | 38.9 seconds |
| VHDL after transformation | 63 lines | 11.6 seconds | 117 lines | 21.5 seconds | 137 lines | 2.4 seconds | 556 lines | 50.6 seconds |
| VHDL after scheduling (cycle accurate) | 78 lines | 37.7 seconds | 305 lines | 42.9 seconds | 173 lines | 3.1 seconds | 1301 lines | 87.4 seconds |
| VHDL structural at RTL level | 353 lines | 210.6 seconds | 1068 lines | 198.9 seconds | 507 lines | 56.9 seconds | 7123 lines | 836 seconds |
| VHDL structural at gate level | 1277 lines | 872.0 seconds | 6234 lines | 3544 seconds | 2767 lines | 1925 seconds | 53219 lines | 3 hours |